REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-98-

0321

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** 26 February 1997	**3. R[illegible]** Final/Technical, 1 Jan 1993–31 Dec 1996

4. TITLE AND SUBTITLE
Scalability in Neural Network Learning and Computation

5. FUNDING NUMBERS
F49620–93–1–0100

6. AUTHOR(S)
Dr. Ian Parberry

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of North Texas, Denton, TX 76203

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
110 Duncan Avenue, Suite B115
Bolling AFB, DC 20332

nm

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Unlimited

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Progress has been made in six topics in the area of computational complexity of neural networks.

- The loading problem for analog neural networks with only 6 nodes is $\mathcal{NP}$-complete.
- Some foundational results on linear threshold functions with Boolean inputs have been extended to real-valued inputs.
- Håstad's lower bound on the dynamic range of weights for linear threshold functions has been improved slightly.
- Theoretical and experimental results have shown that Hopfield nets are inferior to classical sequential and parallel algorithms for the knight's tour problem, a special case of the Travelling Salesperson Problem.
- The problem of finding stable states is $\mathcal{PLS}$-complete even for some very simple and restrictive classes of Hopfield nets.
- We have constructed n-node planar Hopfield networks that take time $2^{n/3}$ to converge sequentially, and $2^{n/7}$ to converge in parallel.

DTIC QUALITY INSPECTED 3

14. SUBJECT TERMS
computers, neural networks, optimization,
Hopfield networks

15. NUMBER OF PAGES
10

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRAC**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)

19980421 154

Scalability in Neural Network Learning and Computation

Ian Parberry
Dept. of Computer Sciences
University of North Texas
P.O Box 13886
Denton, TX 76203-6886

26 February 1997

Final Report for Period 1 January 1993 to 31 December 1996

Prepared for

Major David R. Luginbuhl
AFOSR/NM
110 Duncan Avenue Suite B115
Bolling AFB DC 20332-0001

Scalability in Neural Network Learning and Computation: Final Technical Report

Ian Parberry*
Department of Computer Sciences
University of North Texas

February 26, 1997

1 Summary

We have made progress in the following areas

1. *The Loading Problem:* The proof that the loading problem for analog neural networks with only 6 nodes is $\mathcal{NP}$-complete, first announced by the author in 1992, was completed early in 1993.

2. *Threshold Functions with Real-Valued Inputs:* Foundational results on linear threshold functions with Boolean inputs have been extended to real-valued inputs.

3. *Dynamic Range of Weights:* Håstad's lower bound on the dynamic range of weights for linear threshold functions has been improved slightly.

4. *Hopfield Networks for Knight's Tours:* Takefuji's a Hopfield network for knight's tours was analyzed theoretically and experimentally in comparison with conventional algorithms. We have proven a new lower bound for the number of knight's tours on a generalized chessboard, which throws further light on why Takefuji's Hopfield network for knight's tours works as well as it does.

5. *$\mathcal{PLS}$-completeness of the stable state problem for Hopfield nets:* It has been proved that the problem of finding stable states for some restricted classes of Hopfield nets is just as difficult as the general problem.

6. *Exponential lower bounds for planar Hopfield networks:* We have constructed planar Hopfield networks that take time $2^{n/3}$ to converge under sequential operation, and $2^{n/7}$ to converge in parallel operation.

*Author's address: Department of Computer Sciences, University of North Texas, P.O. Box 13886, Denton, TX 76203–6886, U.S.A. Electronic mail: `ian@cs.unt.edu`.

2 Statement of Work

The Principal Investigator and his Research Assistant shall conduct research into the scalability of neural networks. The research shall analyze the resource requirements of neural network computation and learning as a function of the size of the input domain. The project will investigate the general topic of scalability in neural networks, including the specific areas of research listed below and any related subjects whose interest and relevance are uncovered in the course of this research.

1. The researchers shall investigate whether the loading problem for analog neural networks with a small fixed number of nodes is $\mathcal{NP}$-complete.
2. A hitherto unexplored complexity class, called $\mathcal{ATC}^0$, shall be investigated. This class consists of the problems that can be solved by a neural network with a fixed number of layers and a polynomial number of nodes, in which the first layer of nodes compute linear threshold functions and nodes in the remaining layers compute conjunction and disjunction. Open questions about the relationships between $\mathcal{ATC}^0$ and the well-studied complexity classes $\mathcal{AC}^0$ and $\mathcal{TC}^0$ shall be explored.
3. The problem of determining improved upper and lower bounds on the size of weights in neural networks shall be investigated.
4. Open problems concerning the running time of Hopfield networks used to solve local optimization problems shall be investigated. The theory of $\mathcal{PLS}$ completeness shall be used to to determine whether Hopfield networks for local optimization problems shall require running time that is exponential in the number of inputs.

3 Status of Research

Progress has been made in six areas: the complexity of the loading problem, properties of threshold functions with real-valued inputs, lower bounds for the dynamic range of weights in linear threshold functions, Hopfield networks for knight's tours, Hopfield networks for local optimization, and lower bounds for Hopfield network convergence. Each is discussed below in a separate subsection.

3.1 The Loading Problem

Judd [10, 11, 12, 13] has shown that the problem of loading simple tasks onto neural networks with a fixed architecture is $\mathcal{NP}$-complete, which implies that there is quite likely to be no fast general-purpose learning algorithms even for quite simple architectures and node function sets. Surprisingly, it was shown by Blum and Rivest [1, 2] that the loading problem is $\mathcal{NP}$-complete even for networks consisting of only 3 nodes when the node function set is the set of linear threshold functions. One limitation of the result of Blum and Rivest is that it holds only for node function sets that are *exactly* linear threshold functions. Judd's techniques work for any node function set that includes $\mathcal{AC}_1^0$, and hence apply to analog neural networks, but he has no results for a fixed number of nodes. Our new result is as general as Judd's, but has a fixed node bound. Specifically, we show that the loading problem for a 6-node neural network with node function set $\mathcal{AC}_1^0$ is $\mathcal{NP}$-complete, and deduce deduce that the loading problem for a 6-node analog neural network is $\mathcal{NP}$-hard.

How does one interpret the meaning of $\mathcal{NP}$-completeness results for the loading problem? They imply that any learning algorithm that takes as input a set of tasks and a fixed architecture runs the risk of taking exponential time in the worst case even for architectures drawn from quite innocuous architecture classes. This can be avoided either by limiting the node function set and choosing specific architectures for which the loading problem is not intractable, by allowing the architecture to change during learning, or by only loading task sets that do not cunningly encode $\mathcal{NP}$-complete problems. Results such as ours indicate that even very simple architectures and node function sets can have task sets that encode $\mathcal{NP}$-complete problems. This can be a major pitfall for the unwary neural network designer, and may explain why some learning algorithms scale badly.

3.2 Threshold Functions with Real-Valued Inputs

Discrete linear threshold functions are popular as node functions in discrete neural networks. While there is a fairly substantial body of foundational results on linear threshold functions with Boolean inputs, the corresponding body of work for linear threshold functions with real inputs is not so well-studied. We have shown that over the real domain, all weight-sets of a given linear threshold function are a real multiple of each other. In contrast, it is also shown that every separable linear threshold function over an infinite but bounded domain has an integer weight-set. The latter result is applied to extend the upper bound on the weights in an integer weight-set, and the running time and mistake bounds of the perceptron learning algorithm from the Boolean domain to all finite real domains.

3.3 Dynamic Range of Weights

Håstad [17] has shown that there exist linear threshold functions that requires weights approximately $n^{n/2-n}$. We have, by a careful analysis, tightened the lower bound to the following:

$$\frac{n^{(n-\log n-1)/2}e^{\gamma(\log^2 n+3\log n+8)}(\log^2 n-\log n+1)}{e^{8\gamma n^{\beta}}2^{n-1}}$$

where $\beta = \log(3/2) \approx 0.585$, $\gamma = \ln(4/3) \approx 0.288$.

3.4 Hopfield Networks for Knight's Tours

We began our study of the running time required by Hopfield nets for solving optimization problems by embarking on experimental work. A *knight's tour* is a series of moves made by a knight visiting every square of an $n \times n$ chessboard exactly once. The *knight's tour problem* is the problem of constructing such a tour, given n. The knight's tour problem is interesting because it is a restricted case of the travelling salesperson problem. Takefuji and Lee [19] (reproduced apparently verbatim in Takefuji [18, Chapter 7]) recently proposed a Hopfield-style network for the knight's tour problem. We have compared their algorithm with existing algorithms, both sequentially and in parallel, both for producing single tours and producing multiple tours. We observed that:

1. The Hopfield network was too slow to use for boards larger than 26×26.
2. A random walk algorithm that combines two algorithms invented by Euler (1759) and Warnsdorf (1800s) was tested. It was found to be practical for boards up to 66×66.
3. A divide-and-conquer algorithm quickly gave solutions for boards of size up to 1000×1000.
4. Comparative running times can be seen in Figure 1.

We can conclude that:

1. Experimental evidence has shown that both the Hopfield net and the random walk algorithm run in exponential time. Even so, the random walk algorithm is vastly superior.

2. Theoretical analysis has shown that the divide-and-conquer algorithm runs in linear time. Hence the divide-and-conquer algorithm is far preferable to the other two.
3. In this instance, neural networks don't scale as well as conventional algorithms.

Parallel speedup of the Hopfield network cannot reduce the exponential running time significantly unless exponentially many processors are used. In contrast, the divide-and-conquer algorithm can be implemented in optimal time (that is, proportional to the diameter of the interconnection pattern) using polynomially many processors on many popular architectures. This includes:

- time $O(n^2/p)$ by a bounded degree network with p processors for all $p = O(n^2/\log n)$,
- time $O(n^2/p^2)$ by a $p \times p$ mesh for all $p \leq n^{2/3}$,
- $O(1)$ time on an $n \times n$ mesh with multiple CREW buses, and
- $O(1)$ time on a CREW PRAM with $O(n^2)$ processors.

Similar research has shown that the divide-and-conquer algorithm is much better than the Hopfield network and the random walk algorithm at finding large numbers of tours. We were able to find a lower bound of $\Omega(1.1259^{n^2})$ on the number of knight's tours on an $n \times n$ board for all even $n \geq 6$, and a stronger lower bound of $\Omega(1.2484^{n^2})$ when n is of the form 6×2^k for some $k \in \mathbb{N}$.

Neural networks research has progressed to the point at which the mere fact of existence of a neural network solution for a problem is no longer interesting. Long term interest in neural network research can only be sustained if neural networks can compete with conventional algorithm design techniques. A new algorithm is significant if it can be demonstrated beyond reasonable doubt to be more practical, more effective, or more efficient than existing techniques. We have found that this is not the case for a neural network for the knight's tour problem, neither for finding single tours sequentially or in parallel, nor for finding bounds on the number of tours.

We have worked with Olaf Kyek and Ingo Wegener at the University of Dortmund to show that for all even $n \geq 12$, the number of knight's tours on an $n \times n$ chessboard is

- at least 1.1646^{n^2},
- at most 4^{n^2},
- asymptotically $\Omega(1.3535^{n^2})$.

This indicates that knight's tours are reasonably dense, but decrease exponentially as n approaches infinity. This helps explains why the performance of Takefuji's neural network decreases so rapidly as n increases.

3.5 Hopfield Nets for Local Optimization

$\mathcal{PLS}$-completeness is the analog of $\mathcal{NP}$-completeness for local optimization problems. A local optimization problem that is $\mathcal{PLS}$-complete is considered very unlikely to have a polynomial time algorithm. $\mathcal{PLS}$-completeness was invented by Johnson, Papadimitriou, and Yannakakis [9]. Recently, Papadimitriou, Schäffer, and Yannakakis [16] showed that the problem of finding stable states in a Hopfield network in $\mathcal{PLS}$-complete. This is both good news and bad news for proponents of Hopfield nets. It shows that they are powerful computational models that are capable of encoding problems that are difficult for conventional computers, but also shows that they are unlikely to run in polynomial time.

The argument of Papadimitriou, Schäffer, and Yannakakis [16] shows that the stable state problem for Hopfield nets is $\mathcal{PLS}$-complete in general, but does not address the case of restricted Hopfield nets. The Hopfield nets found in the literature typically have restrictions on the underlying graph. We have preliminary proofs of the following results. The stable state problem for Hopfield nets is still $\mathcal{PLS}$-complete even for:

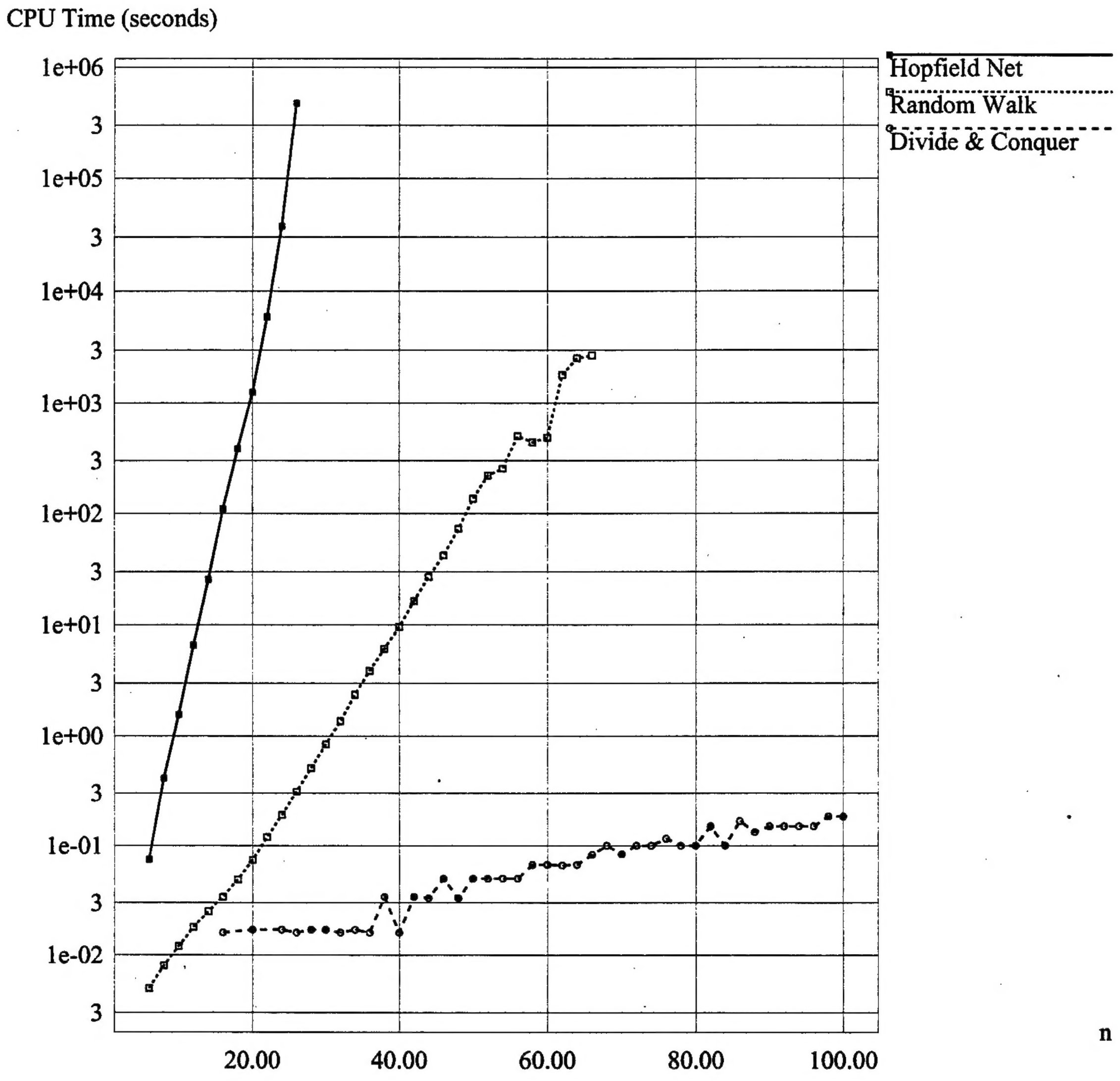

Figure 1: Running times for the Hopfield network, random walk algorithm, and the divide-and-conquer algorithm on an $n \times n$ board. Note that the time scale is logarithmic.

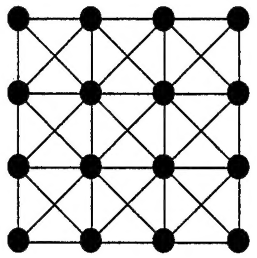

Figure 2: A grid graph.

1. Hopfield nets whose interconnection pattern is bipartite, that is, the neurons are divided into two sets L and R such that if there is a connection between neurons v and w, then $v \in L$ and $w \in R$, or vice-versa.
2. Hopfield nets of degree 3, that is, every neuron is connected to at most 3 other neurons.
3. Hopfield nets whose interconnection pattern is a very simple grid graph with crossovers in each grid element (see Figure 2).
4. Hopfield nets whose interconnection pattern is one of a class of graphs used heavily in parallel computation, including the hypercube, the butterfly, the cube-connected cycles, and the shuffle-exchange (see, for example, Leighton [14]). Hopfield nets whose interconnection pattern is the dual of the knight's graph, used by Takefuji and Lee [19].

Result (1) is proved by adapting a technique due to Bruck and Goodman [4, 5]. Results (2–4) are proved using a graph embedding technique. It is first shown that the stable state problem for Hopfield nets with interconnection pattern from an arbitrary class of graphs C is $\mathcal{PLS}$-complete if an arbitrary graph G can be embedded in a graph from C with congestion 1, load 1, and polynomial expansion. (An embedding of a graph $G_1 = (V_1, E_1)$ into $G_2 = (V_2, E_2)$ is a function $f : V_1 \rightarrow V_2$ such that for all $(v, w) \in E_1$, there exists a path between $f(v)$ and $f(w)$ in E_2. The *congestion* is the maximum number of such paths that uses any single edge, the *load* is the maximum number of vertices in V_1 that have the same image under f, and the *expansion* is the size of V_2 as a function of the size of V_1.) We are then able to show that suitable embeddings exist for the required classes of graphs.

$\mathcal{PLS}$-completeness results for Hopfield nets can be interpreted in two ways. An optimistic interpretation is that they identify the problem of finding stable states in a Hopfield network as a key problem in the theory of local optimization. A pessimistic interpretation is that it is unlikely that a fast algorithm can be found for finding stable states in Hopfield networks, since such an algorithm would provide fast algorithms for problems that have resisted such attempts for a long period of time. We have provided strong evidence that there are no polynomial time algorithms for finding stable states in some very simple classes of Hopfield nets, and in particular that each class includes Hopfield networks that do not converge in polynomial time.

3.6 Lower Bounds for Hopfield Network Convergence

A planar Hopfield network is one whose interconnection graph is planar, that is, can be drawn on the Euclidean plane without crossing edges. Haken and Luby [8] describe a planar Hopfield network that provably takes exponential time to converge. Yet there exists an algorithm for the stable state problem that runs in polynomial time on *all* Hopfield networks. The proof follows from the fact that the maximal cut in a planar graph can be found in polynomial time (see, for example, Hadlock [7]), combined with results of Papadimitriou, Schäffer, and Yannakakis [16]. Thus, we have proved that conventional computers are exponentially faster than planar Hopfield networks in sequential mode.

We further considered the question of how long the convergence time for a planar Hopfield network can be, in both sequential and parallel operation. In the case of sequential updates, Haken and Luby [8] construct a planar Hopfield network on n nodes that takes time $2^{n/7}$ to converge. Goles and Martinez [6], building on the work of Orponen [15], extended this to $2^{n/6}$ in the nonplanar case. Using different techniques, we were able to prove a lower bound of $2^{n/3}$ in the planar case, thus improving substantially on both results.

In the case of parallel updates, Goles and Martinez [6], and Bruck [3] separately described a Hopfield network that takes time $2^{n/3}$ to converge. There were no corresponding results for planar Hopfield networks in parallel mode. We have constructed a planar Hopfield network that takes time $2^{n/7}$ to converge.

4 Publications

1. I. Parberry, "Training a 6-node Analog Neural Network is $\mathcal{NP}$-hard". Technical Report CRPDC–93–1, Center for Research in Parallel and Distributed Computing, Dept. of Computer Sciences, Univ. of North Texas, Jan. 1993. Submitted January 1993 to *Journal of Complexity.*

2. I. Parberry, "On Linear Threshold Functions with Real Inputs", Technical Report CRPDC–93–9, Center for Research in Parallel and Distributed Computing, Dept. of Computer Sciences, Univ. of North Texas, June 1993.

3. I. Parberry, "Circuit Complexity and Neural Networks", MIT Press, 1994.

4. I. Parberry, "Structural Complexity and Neural Networks", "The Handbook of Brain Theory and Neural Networks", (Michael Arbib, Ed.), pp. 945–948, MIT Press, 1995.

5. I. Parberry, "Algorithms for Touring Knights", Technical Report CRPDC–94–7, Center for Research in Parallel and Distributed Computing, Dept. of Computer Sciences, Univ. of North Texas, May 1994.

6. I. Parberry, "Scalability of a Neural Network for the Knight's Tour Problem", *Neurocomputing* Vol. 12, pp. 19–34, 1996.

7. I. Parberry, "A Divide-and-Conquer Algorithm for the Knight's Tour Problem", Accepted December 1995 to *Discrete Applied Mathematics.*

8. O. Kyek, I. Parberry, and I. Wegener. "Bounds on the Number of Knight's Tours", Accepted January 1996 to *Discrete Applied Mathematics.*

9. I. Parberry and H.-L. Tseng. "Are Hopfield Networks Faster Than Conventional Computers?", to appear in the Proceedings of the 10th Conference on Neural Information Systems — Natural and Synthetic, Denver, Colorado, Nov. 1996.

10. I. Parberry and H.-L. Tseng. "Exponential Lower Bounds for Hopfield Networks", in preparation.

5 Personnel

Name	*Title*	*Affiliation*
Ian Parberry	Principal Investigator	Professor
Keith Dean	Research Assistant	Graduate Student
B. Chitturi	Research Assistant	Graduate Student
Hung-Li Tseng	Research Assistant	Graduate Student

6 Interactions

1. "Are Hopfield Networks Faster Than Conventional Computers?", with Hung-Li Tseng, Poster and spotlight oral presentation at the 10th Conference on Neural Information Systems — Natural and Synthetic, Denver, Colorado, Nov. 1996.

2. I. Parberry, "The Complexity of Stability in Hopfield Networks", Post Meeting Workshop on "Optimization Problem Solving in Neural Networks", 1995 Conference on Neural Information Processing Systems — Natural and Synthetic, Vail, Colorado, Dec. 1995.

3. I. Parberry, "Algorithms for Touring Knights", Invited Poster, Center for Network Neuroscience, 1994 Center Symposium, University of North Texas, March 1994.

4. I. Parberry, "Algorithms for Touring Knights". Colloquium, Dept. of Computer Science, Johns Hopkins Univ., Baltimore, MD, June 1994.

5. I. Parberry, "Algorithms for Touring Knights". Workshop on Algorithmic Research in the Midsouthwest, Austin, TX, Nov. 1994.

6. I. Parberry, "Algorithms for Touring Knights". Invited talk, Workshop on "Neural Network Methods for Optimization Problems", 1993 Conference on Neural Information Processing Systems — Natural and Synthetic, Vail, Colorado, Dec. 1993.

7. I. Parberry, "Algorithms for Touring Knights". Invited talk, DIMACS Workshop on "Parallel Algorithms: From Solving Combinatorial Problems to Solving Grand Challenge Problems", Rutgers University, Piscataway, New Jersey, Nov. 1993.

References

[1] A. Blum and R. L. Rivest. Training a 3-node neural network is NP-complete. In *Neural Information Processing Systems 1*, pages 494–501. Morgan Kaufmann, 1989.

[2] A. Blum and R. L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5(1):117–127, 1992.

[3] J. Bruck. On the convergence properties of the Hopfield model. *Proceedings of the IEEE*, 78(10):1579–1585, 1990.

[4] J. Bruck and J. W. Goodman. A generalized convergence theorem for neural networks and its applications in combinatorial optimization. In *Proc. IEEE First International Conference on Neural Networks*, volume III, pages 649–656, San Diego, CA, June 1987.

[5] J. Bruck and J. W. Goodman. A generalized convergence theorem for neural networks. *IEEE Transactions on Information Theory*, 34(5):1089–1092, 1988.

[6] E. Goles and S. Martinez. Exponential transient classes of symmetric neural networks for synchronous and sequential updating. *Complex Systems*, 3:589–597, 1989.

[7] F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4(3):221–225, 1975.

[8] A. Haken and M. Luby. Steepest descent can take exponential time for symmetric conenction networks. *Complex Systems*, 2:191–196, 1988.

[9] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? In *26th Annual Symposium on Foundations of Computer Science*, pages 39–42. IEEE Computer Society Press, 1985.

[10] J. S. Judd. Learning in networks is hard. In *Proc. of the First International Conference on Neural Networks*, pages 685–692. IEEE Computer Society Press, 1987.

[11] J. S. Judd. *Neural Network Design and the Complexity of Learning.* PhD thesis, University of Massachusetts, Amherst, MA, 1988.

[12] J. S. Judd. On the complexity of loading shallow neural networks. *Journal of Complexity*, 4:177–192, 1988.

[13] J. S. Judd. *Neural Network Design and the Complexity of Learning.* MIT Press, 1990.

[14] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes.* Morgan Kaufmann, 1992.

[15] P. Orponen. The computational power of discrete hopfield nets with hidden units. *Neural Computation*, 8(2):403–415, February 1996.

[16] C. H. Papadimitriou, A. A. Schäffer, and M. Yannakakis. On the complexity of local search. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 439–445. ACM Press, 1990.

[17] J. Håstad. On the size of weights for threshold gates. Unpublished Manuscript, 1992.

[18] Y. Takefuji. *Neural Network Parallel Computing.* Kluwer Academic Publishers, 1992.

[19] Y. Takefuji and K. C. Lee. Neural network computing for knight's tour problems. *Neurocomputing*, 4(5):249–254, 1992.